Continuous Assurance: Continuous Integration meets Containers Security

Jordan Alexis Caraballo Vega, George Rumney, John Jasen, Jasaun Neff
[1]University of Puerto Rico at Humacao, Department of Mathematics, Humacao, PR
[2]High Performance Computing Branch, NASA Center for Climate Simulation (NCCS), NASA Goddard Space Flight Center, Mail Code 606.2, Greenbelt, MD, USA

CI at the NCCS

Continuous Integration (CI) is a software development technique where developers integrate code into one or more shared repositories on a frequent basis. CI key principles reside in revision control, build automation, and automated testing. It has the capacity of decreasing debugging time while increasing development productivity.

Figure 1. Default CI workflow. Phases in their respective order include: developers committing to a version control repository, pulling the commit, building the software project, and testing the product in an environment. At the end, notifications are sent to the corresponding developers.

Based on security concerns, the NCCS has refined the idea of enhancing CI with Continuous Assurance (CA) techniques. The aim of this project is to research and build a reinforced CI/CA infrastructure able to significantly reduce security risks and monitor the build processes. Added features will include voting to approve/disapprove the associated workflows and any changes.

Security Risks Involved in CI

While in an ideal world CI itself sounds like the perfect approach to boost the productivity of software teams, its exposure to the network makes it vulnerable to various kinds of malicious attacks. Some of these common vectors include:

- Remote Code Execution (RCE)
- Deserialization
- Cross-site request forgery (CSRF)
- Information Disclosure

OWASP Open Web Application Security Project

Additionally, these software projects rely on unit tests that in the end lack the ability of detecting bad programming practices that could lead to further security risks. The idea behind this project is to strategically implement security standards in each one of the CI development phases (Figure 2).

Figure 2. CI/CA reinforced workflow. One or more security tests will be performed after each phase. All projects must pass an initial vetting process that will be approved by the Security Team. After the workflow is approved, the process can continue automatically until a major change or new unapproved vulnerabilities are found.

By enforcing the execution of several security tools in each phase, teams will be able to find and fix issues and vulnerabilities early in the development process. This approach will also prevent big architectural flaws that are sometimes introduced in software projects that are not integrated regularly. These security tests will cover the majority of the security vectors involved in software and systems development.

Methods: Continuous Assurance Taking Over

Software Project Workflow

Java and C++ projects were developed on Git to simulate the interaction of users within the CI infrastructure. An additional Fortran project was cloned from an existing GSFC team.

Bamboo was used as the CI platform. A new project was created for each Git branch, and a default job that gets triggered after any commit to the repository was configured.

Since these software projects were not web applications, the only security test performed was the execution of static source code analysis tools for each programming language. Figure 3 has an example of this workflow. By adding these tools we can prevent inside threats that could come from the source code itself.

Figure 3. Software project workflow. This workflow was tested with and without Bamboo's native agent. From left to right: Bamboo job tasks, build step output, several source code scan tools, and simple metrics regarding the project progress.

Container Images Workflow – CIS & SCAP Compliant

The previous workflow took care of source code vulnerabilities that could be found on the repository, however it did not validate the image where it was being built. Therefore, a workflow to validate container images and to prevent RCE was built.

This process will equip the CI/CA infrastructure with the ability of isolating builds and validating the compliance of operating systems that will simulate production-like images at the NCCS. CIS and SCAP Compliant systems will also satisfy BigFix scans and will guarantee that the container is in optimal condition to be deployed as an agent.

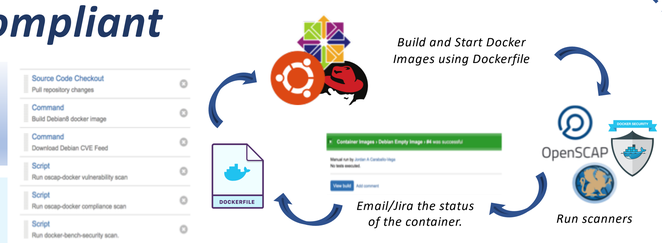

Figure 4. Container Images workflow. A Dockerfile is defined that can download a local or remote image. SCAP compliance and vulnerability tests are ran using oscap-docker, while CIS tests are executed by docker-bench-security.

Containerized Software Application Workflow

This last workflow seeks to integrate user's software projects and NCCS approved containers to simulate the behavior and interaction of the provided software services on a NCCS production system while introducing security tools to validate its integrity.

A Python Django application was written and pushed to Git, while an additional PHP known vulnerable application was taken from another repository. Each project was built on a Docker container, scanned, and vetted. Vulnerabilities found at the Django application were fixed and retested through the CI/CA workflow.

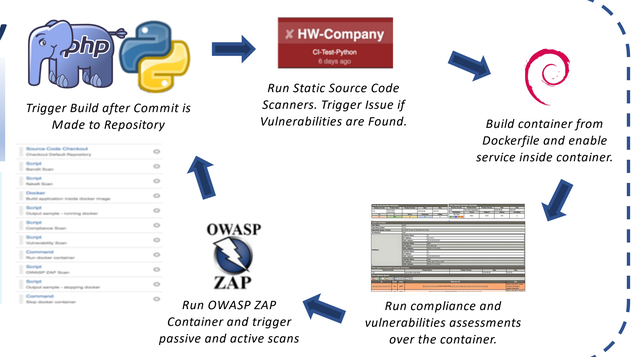

Figure 5. Isolated Software Project workflow. A security tool is implemented at each phase of the workflow. Tools go from source code analysis, to penetration testing. At the end, issues are followed on Jira.

Visualizations and System Metrics

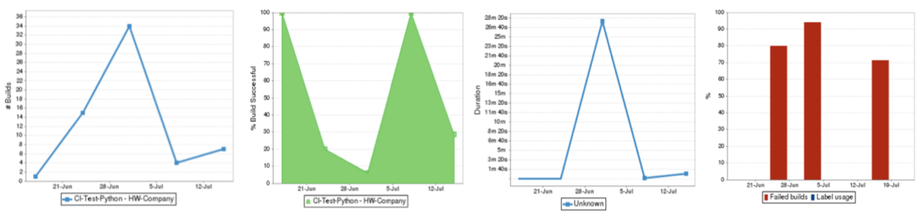

Figure 6. Bamboo Python Project Visualizations. (a) Linear representation of the number of builds as a function of time. (b) Percentage of successful builds as a function of the number of builds through time. (c) Duration of the builds and (d) percentage of failed builds as a function of time.

As a team, it is extremely important to monitor the overall progress of each project. Therefore, each workflow in Bamboo is equipped with system metrics that report the status of each project as a function of time. This feature will enable the notification of management if there are continuous integration issues or security flaws that could be solved or attended. An additional system metrics report will be computed from the infrastructure to measure the use of its resources and together with the computing time that each project is taking.

Conclusion and Remarks

At this time, the NCCS has an infrastructure capable of fulfilling CI/CA needs. Thanks to this approach, it can be concluded that CI workflows need to be reinforced with CA techniques to prevent additional security risks that the execution and integration of software could bring into play. By isolating builds into containers we are able to add one protection layer to the process. These techniques hold the potential to speed up the development of software at the NCCS and could serve as a testing environment for system administrators. Future work will include the integration of new tools and its deployment. The author wants to thank the NASA Minority University Research and Education Scholarship and his mentors for their exceptional contribution. For further references, you may contact the author of this poster.